Explanation-Guided Diagnosis of Machine Learning Evasion Attacks

Abderrahmen Amich(✉) and Birhanu Eshete

University of Michigan, Dearborn, USA
{aamich,birhanu}@umich.edu

Abstract. Machine Learning (ML) models are susceptible to evasion attacks. Evasion accuracy is typically assessed using aggregate evasion rate, and it is an open question whether aggregate evasion rate enables feature-level diagnosis on the effect of adversarial perturbations on evasive predictions. In this paper, we introduce a novel framework that harnesses explainable ML methods to guide high-fidelity assessment of ML evasion attacks. Our framework enables explanation-guided correlation analysis between pre-evasion perturbations and post-evasion explanations. Towards systematic assessment of ML evasion attacks, we propose and evaluate a novel suite of model-agnostic metrics for sample-level and dataset-level correlation analysis. Using malware and image classifiers, we conduct comprehensive evaluations across diverse model architectures and complementary feature representations. Our explanation-guided correlation analysis reveals correlation gaps between adversarial samples and the corresponding perturbations performed on them. Using a case study on explanation-guided evasion, we show the broader usage of our methodology for assessing robustness of ML models.

Keywords: Machine learning evasion · Explainable machine learning

1 Introduction

The widespread usage of machine learning (ML) in a myriad of application domains has brought adversarial threats to ML models to the forefront of research towards dependable and secure ML systems. From image classification [29] to voice recognition [12], from precision medicine [18] to malware/intrusion detection [38] and autonomous vehicles [42], ML models have been shown to be vulnerable not only to training-time poisoning and evasion attacks, but also to model extraction and membership inference attacks [10]. In typical evasion attacks, an adversary perturbs a legitimate input to craft an *adversarial sample* that tricks a victim model into making an incorrect prediction.

Motivation: Prior work has demonstrated adversarial sample-based evasion of ML models across diverse domains such as image classifiers [11,20,30], malware classifiers [6,9,14,26,38], and other domains such as speech and text processing. In the current state-of-the-art, the effectiveness of evasion is typically assessed

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 207–228, 2021.
https://doi.org/10.1007/978-3-030-90019-9_11

through *aggregate evasion rate* by computing the percentage of crafted adversarial samples that lead a model to make evasive predictions. For a ML model f that accepts a d-dimensional input $x = [x_1, ..., x_d]$ to predict $f(x) = y_{true}$, the adversary perturbs x to obtain $x' = [x_1 + \delta_1, ..., x_d + \delta_d]$, where $\delta = [\delta_1, ..., \delta_d]$ represents *pre-evasion perturbations* applied to each feature. When f is queried with x', it produces an evasive prediction $f(x') = y' \neq y_{true}$. The natural question then is whether there exists *correlation between pre-evasion perturbations and the evasive prediction*. Unfortunately, aggregate evasion rate is inadequate to offer fine-grained insights to answer this question. It does not show how much the evasion strategy, through adversarial perturbations, influences individual samples to result in an evasive prediction. We consequently argue that unless one "unpacks" aggregate evasion rate at the resolution of an adversarial sample, it could give false sense of evasion success for it lacks the fidelity at the level of individual features. Such a coarse-grained nature of the aggregate evasion metric can potentially misguide the evaluation of model robustness in the face of adversarial manipulations.

Approach: In this paper, we harness feature-based ML explanation methods and propose an explanation-guided correlation analysis framework for evasion attacks on ML models. Explainable ML techniques [23,33,39,44] interpret predictions returned by a ML model and attribute model's decision (e.g., predicted class label) to feature importance weights. In particular, for each evasive prediction $f(x') = y' \neq y_{true}$, explanation methods such as LIME [39] and SHAP [33] produce *post-evasion explanations* of the form $[x_1 : w_1, ..., x_d : w_d]$, where w_i is the weight of contribution of feature x_i to the evasive prediction y'. To address the lack of detailed insights from aggregate evasion rate, we leverage post-evasion explanations and empirically explore their feature-level correlations with pre-evasion perturbations performed by the adversary. Our key insight is that, since the perturbations are the only manipulations done on the feature-space of an input sample, when the model makes an evasive prediction on a perturbed variant of the input sample, there should exist some correlation between pre-evasion perturbations and post-evasion explanations. Towards systematic assessment of the link between adversarial perturbations and evasive predictions, we propose and evaluate a novel suite of metrics that allow (adversarial) sample-level and (evasion) dataset-level diagnosis of evasion attacks. Our suite of metrics is applicable to any ML model that predicts a class label given an input because, in the design of the metrics, we make no assumptions about the ML task and model architecture. The benefit of the fine-grained diagnosis for a defender is twofold. First, it enables systematic measurement of the strength of correlation between an evasive prediction and feature-level perturbations across diverse classification tasks, model architectures, and feature representations. Second, it allows zooming-in on limitations of feature perturbation strategies to inform pre-deployment adversarial robustness evaluation of ML models.

Note on Scope: This paper is not yet another adversarial sample crafting approach for which problem-space to feature-space mapping is crucial to maintain functional integrity of adversarial samples (e.g., in adversarial malware samples). Our approach rather relies on *feature-space perturbations* performed to craft an

adversarial sample and model output explanations of the same sample. Our goal correlation analysis between perturbations and explanations.

Evaluation Highlights: We evaluate our framework across different classification tasks (image, malware), diverse model architectures (e.g., neural networks, multiple tree-based classifiers, logistic regression), and complementary feature representations (pixels for images, static and dynamic analysis-based features for malware). Our explanation-guided correlation analysis reveals an average of 45% per-model adversarial samples that have low correlation links with perturbations performed on them –indicating the inadequacy of aggregate evasion rate, but the utility of fine-grained correlation analysis, for reliable diagnosis of evasion accuracy. Our results additionally suggest that, although a perturbation strategy evades a target model, at the granularity of each feature perturbation, it can lead to a per-model average of 36% *negative* feature perturbations (i.e., perturbations that contribute to maintain the original true prediction $f_b(x') = y_{true}$). We further evaluate the utility of our framework in a case study on explanation-guided adversarial sample crafting.

Contributions: In summary, this paper makes the following contributions:

- *Explanation-guided diagnosis of evasion attacks.* To improve the sole reliance of evasion assessment on aggregate evasion rate, we propose an *explanation-guided correlation analysis* framework at the resolution of individual features. To that end, we introduce a novel *suite of correlation analysis metrics* and demonstrate their effectiveness at pinpointing adversarial examples that indeed evade a model yet exhibit loose correlation with perturbations performed to craft them.
- *Comprehensive evaluations.* In malware classification and image classification, we conduct extensive evaluations across diverse model architectures and feature representations, and synthesize interesting experimental insights that demonstrate the utility of explanation-guided correlation analysis.
- *Further case study.* We conduct a case study using *pre-perturbation feature direction analysis* to guide evasion strategies towards crafting more accurate adversarial samples *correlated* with their evasive predictions.

2 Background: ML Evasion and Explanation Methods

In this section, we introduce ML evasion attacks and ML explanation methods.

2.1 ML Evasion Attacks

Adversarial Sample Crafting. Given a deployed ML model (e.g., malware classifier, image classifier) with a decision function $f : X \rightarrow Y$ that maps an input sample $x \in X$ to a true class label $y_{true} \in Y$, then $x' = x + \delta$ is called an *adversarial sample* with an *adversarial perturbation* δ if: $f(x') = y' \neq y_{true}, ||\delta|| < \epsilon$, where $||.||$ is a distance metric (e.g., one of the L_p norms) and ϵ is the maximum allowable perturbation that results in misclassification while preserving semantic integrity of x. Semantic integrity is domain and/or task specific. For instance,

in image classification, visual imperceptibility of x' from x is desired while in malware detection x and x' need to satisfy certain functional equivalence (e.g., if x was a malware pre-perturbation, x' is expected to exhibit maliciousness post-perturbation as well). In *untargeted* evasion, the goal is to make the model misclassify a sample to any different class (e.g., for a roadside sign detection model: misclassify red light as any other sign). When the evasion is *targeted*, the goal is to make the model to misclassify a sample to a specific target class (e.g., in malware detection: misclassify malware as benign).

Evasion attacks can be done in *white-box* or *black-box* setting. Most gradient-based evasion techniques [11,20,30,34] are white-box because the adversary typically has access to model architecture and parameters/weights, which allows to query the model directly to decide how to increase the model's loss function. In recent years, several white-box adversarial sample crafting methods have been proposed, specially for image classification tasks. Some of the most notable ones are: Fast Gradient Sign Method (FGSM) [20], Basic Iterative Method (BIM) [30], Projected Gradient Descent (PGD) method [34], and Carlini & Wagner (CW) method [11]. Black-box evasion techniques usually start from some initial perturbation δ_0, and subsequently probe f on a series of perturbations $f(x + \delta_i)$, to craft a variation of x that evades f (i.e., misclassified to a label different from its original). In malware classifiers, while *gradient-based* methods have been widely adopted both in white-box and black-box settings, two other strategies also standout for evasion in a black-box setting. The first one is called *additive* because it appends adversarial noise (e.g., no-op bytes) to the end of a sample (e.g., Windows PE) to preserve original behavior [21,28]. The second class of methods uses *targeted and constrained manipulations* after identifying regions in the PE that are unlikely to be mapped to memory [49].

One of the challenges for state-of-the-art adversarial sample crafting methods is the lack/impossibility of mapping of feature-space perturbations to the problem space. Such a mapping and reversibility between the two spaces is essential in domains where the functionality of an adversarial sample needs to be preserved post-perturbation [24]. A recent work by Pierazzi et al. [37] proposes formulations and shows promising experimental results towards the feasibility of crafting evasive malware samples with real-world consequences. As discussed in Sect. 1, problem space perturbations are out of this paper's scope.

2.2 ML Explanation Methods

ML models have long been perceived as black-box in their predictions until the advent of explainable ML [33,39,44], which attribute a decision of a model to features that contributed to the decision. This notion of attribution is based on quantifiable contribution of each feature to a model's decision.

ML explanation is usually accomplished by training a substitute model based on the input feature vectors and output predictions of the model, and then use the coefficients of that model to approximate the importance and *direction* (class label it leans to) of the feature. A typical substitute model for explanation is of the form: $s(x) = w_0 + \sum_{i=1}^{d} w_i x_i$, where d is the number of features, x is

the sample, x_i is the i^{th} feature for sample x, and w_i is the contribution/weight of feature x_i to the model's decision. While ML explanation methods exist for white-box [45,47] or black-box [23,33,39] access to the model, in this work we consider ML explanation methods that have black-box access to the ML model, among which the notable ones are LIME [39], SHAP [33] and LEMNA [23]. Next, we briefly introduce these explanation methods.

LIME and SHAP. Ribeiro et al. [39] introduce LIME as one of the first model-agnostic black-box methods for locally explaining model output. Lundberg and Lee further extended LIME by proposing SHAP [33]. Both methods approximate the decision function f_b by creating a series of l perturbations of a sample x, denoted as $x'_1, ..., x'_l$ by randomly setting feature values in the vector x to 0. The methods then proceed by predicting a label $f_b(x'_i) = y_i$ for each x'_i of the l perturbations. This sampling strategy enables the methods to approximate the local neighborhood of f_b at the point $f_b(x)$. LIME approximates the decision boundary by a weighted linear regression model as: $_{g \in G} \sum_{i=1}^{l} \pi_x(x'_i)(f_b(x'_i) - g(x'_i))^2$, where G is the set of all linear functions and π_x is a function indicating the difference between the input x and a perturbation x'. SHAP follows a similar approach but employs the SHAP kernel as weighting function π_x, which is computed using the *Shapley Values* [43] when solving the regression. Shapley Values are a concept from game theory where the features act as players under the objective of finding a fair contribution of the features to the payout –in this case the prediction of the model.

LEMNA. Another black-box explanation method designed work well for non-linear models is LEMNA [23] that uses a mixture regression model for approximation, i.e., a weighted sum of K linear models defined as: $f(x) = \sum_{j=1}^{K} \pi_j(\beta_j.x + \epsilon_j)$, where the parameter K specifies the number of models, the random variables $\epsilon = (\epsilon_1, ..., \epsilon_K)$ originate from a normal distribution $\epsilon_i \sim N(0, \sigma)$ and $\pi = (\pi_1, ..., \pi_K)$ holds the weights for each model. The variables $\beta_1, ..., \beta_K$ are the regression coefficients and can be interpreted as K linear approximations of the decision boundary near $f_b(x)$.

3 Explanation-Guided Evasion Diagnosis Framework

In this section, we present our explanation-guided correlation analysis methodology. Table 1 describes notations used here and in the rest of the paper.

3.1 Overview

As described in Sect. 1, the effectiveness of an evasion method is typically assessed using aggregate evasion accuracy. While aggregate evasion quantifies the overall success of an evasion strategy, it fails to offer sufficient insights. It does not show how the evasion mechanism influences individual samples to result in evasive predictions. We argue that, unless one examines evasion success at the resolution of each adversarial sample, aggregate evasion rate could give

Table 1. Notations.

Notation	Brief description
X_b	Training set of black-box model f_b
X_e	Evasion set disjoint with X_b
X'_e	Adversarial counterpart of X_e
X_s	Training set of explanation model f_s
$x \in X_e$	Sample in evasion set
$x' \in X'_e$	Adversarial variant of x
$Y = \{y_1, ..., y_k\}$	Set of classes (labels)
$x = [x_1, ..., x_d]$	d-dimensional feature vector of sample x
$W_{x,y_i} = \{w_1, ..., w_d\}$	Feature weights (explanations) of a sample x toward the class y_i
$pos(x, y_i)$	Number of features in x *positive* to the prediction $f_b(x) = y_i$
$neg(x, y_i)$	Number of features in x *negative* to the prediction $f_b(x) = y_i$
$neut(x, y_i)$	Number of features in x *neutral* to the prediction $f_b(x) = y_i$
$P(x')$	Number of perturbed features in x'
τ	Threshold to decide highly correlated adversarial samples

false sense of adversarial success for it lacks feature-level fidelity of perturbations that result in an adversarial sample. To address the stated lack of fidelity in aggregate evasion accuracy, we systematically explore how ML explanation methods are harnessed to assess feature-level correlations between pre-evasion adversarial perturbations and post-evasion explanations.

Figure 1 shows an overview of our explanation-guided correlation analysis framework. Given an evasion set X'_e of adversarial samples, our framework enables correlation analysis both at the sample-level (for each $x' \in X'_e$ at the granularity of each perturbed feature) and at the evasion dataset-level ($\forall x' \in X'_e$). Intuitively, given a decisive feature (obtained via ML explanations) of an evasive sample ($f_b(x') \neq y_{true}$), for such a feature to be considered the cause of (correlated to) the evasion, there needs to be a corresponding feature that was perturbed in the original sample x. By repeating the correlation of each decisive feature with its perturbed counterpart, our sample-level correlation analysis establishes empirical evidence that links an evasive prediction with its cause.

More precisely, our correlation analysis is performed by harnessing the *post-evasion features directions* ("2. Explanation" in Fig. 1) of adversarial samples ("1. Evasion" in Fig. 1). First, we explore the feature directions of the *pre-evasion perturbations* to obtain an assessment of the contribution of each feature perturbation to the attack (i.e., *feature-level assessment*). Second, we use those results to zoom-out to a *sample-level assessment* (Sect. 3.3). Finally, we move to the higher level of the whole evasion dataset to obtain an overall assessment of the evasion attack (Sect. 3.4).

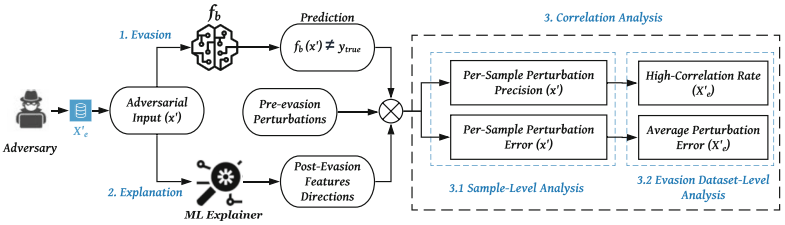

Fig. 1. Explanation-guided correlation analysis framework.

Conducting such fine-grained correlation analysis has two key benefits. Firstly, it verifies whether evasion can be attributed to the adversarial perturbations employed on the sample, and, in effect, performs diagnosis on aggregate evasion accuracy. Secondly, it provides visibility into how sensitive certain samples and/or features are to adversarial perturbations, which could inform robustness assessment of ML models in the face of evasion attacks.

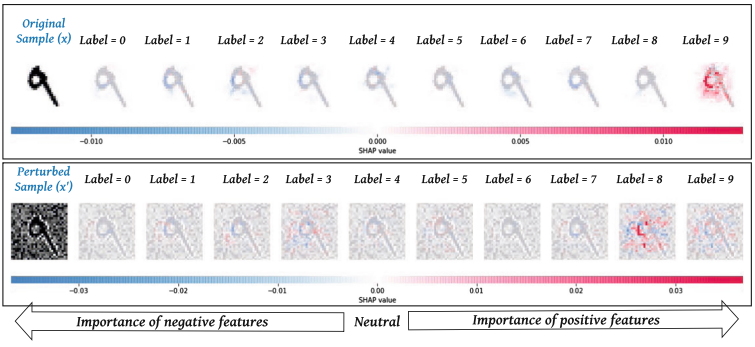

Fig. 2. A comparative illustration of pre-perturbation and post-perturbation explanations using SHAP [33] on a test sample from the MNIST [31] dataset.

3.2 Post-Evasion Feature Direction

In a typical classification task, for an input sample x, $f_b(x) = y_i \in Y = \{y_1, ..., y_k\}$, where Y is the set of k possible labels. For example, in the multi-class handwritten digit recognition model of the MNIST [31] dataset, the input is an image of a handwritten digit and the label is one of the 10 digits (i.e., $Y = \{0, .., 9\}$ where $k = 10$). In the malware detection domain, the typical model is a binary classifier (i.e., $Y = \{\texttt{Benign}, \texttt{Malware}\}$ where $k = 2$). Next, we use MNIST as an illustrative example to describe post-evasion feature direction.

Explanations returned from ML explanation methods reveal the *direction* of each feature. For each class $y_i \in Y$ and an adversarial sample x', a ML explanation method returns a set of *feature weights* $W_{x',y_i} = \{w_1, .., w_d\}$ where

w_j reflects the importance (as the magnitude of w_j) and the *direction* (as the sign of w_j) of the feature x'_j towards the prediction $f_b(x') = y_i$. Depending on the sign of w_j, feature x'_j can be *positive*, *negative*, or *neutral* with respect to the prediction $f_b(x') = y_i$. When $w_j > 0$, we say x'_j is positive to (directed towards) y_i. Conversely, when $w_j < 0$, x'_j is negative to (directed away from) y'_i. When $w_j = 0$, we say x'_j is neutral to y_i (does not have any impact on the prediction decision). In case of binary classification ($k = 2$), if x'_j is not directed to the label y_1 (`Benign` for malware detection) and is not neutral, then x'_j can only be directed to the other label y_2 (`Malware`) and vice versa. To illustrate how we leverage feature direction in our analysis, next we describe a concrete example from the MNIST [31] handwritten digit recognition model.

In Fig. 2, the upper box shows SHAP [33] pre-evasion feature explanations of a correct prediction on an image of "9"(i.e., $f_b(x) = 9$). The lower box shows post-evasion feature explanations of the misclassification $f_b(x') \neq 9$ using an adversarial variant $x' \leftarrow x + \delta$. Each column (i.e., "$Label = y_i$"; $y_i \in \{0, ..., 9\}$) represents the feature directions for the possibility of a prediction $f_b(x) = y_i$ (upper box) and $f_b(x') = y_i$ (lower box). The color codes are interpreted as follows: given an explanation, pink corresponds to positive features while blue corresponds to negative features. The intensity of either color (pink or blue) is directly proportional to the feature weight towards the prediction. Neutral features are represented with white. For instance, focusing on the correct prediction label $y_{true} = 9$ in the upper-box, we notice a large concentration of pink features which positively contribute to the predicted label (9).

Our approach primarily relies on post-evasion explanations (lower box in Fig. 2) and we observe that feature importance weights vary for each studied label as a potential prediction $f_b(x') = y_i \in \{0, ..., 9\}$. When the prediction $f_b(x') = 8$ (image below 'Label = 8' in lower box), the explanations show that most features are *positive* (directed to label 8), which explains the change of the prediction label from 9 to 8. Examining the colors, we realize that most features that were directed to label 9 in the pre-perturbation explanations have become either neutral to the prediction $f_b(x') = 9$ or are positive towards $f_b(x') = 8$. It is noteworthy that some perturbed features are oriented to the original label 9 (notice pink pixels in the image below 'Label = 9' in lower box). Such observations suggest that even though the attack is successful (i.e., $f_b(x') = 8 \neq 9$), the effectiveness of each single feature perturbation is not guaranteed to result in an evasive prediction. Thus, the evasion success may not always be correlated with each feature perturbation the adversary performs on the original sample. We, therefore, argue that a perturbation strategy that produces many features that are uncorrelated with the misclassification might perform poorly on other feature representations (e.g., colored or not centered images in image classification) or other feature types (e.g., static vs. dynamic features in malware detection) which reflects a potential limitation of the stability of a perturbation method. Next, we introduce novel sample-level metrics that capture the fine-grained assessment that leverages post-evasion explanations. We refer to Table 1 for the feature direction-related notations. Our focus will be on the *post-perturbation feature directions* of an evasive sample x' and we suppose that its original prediction (pre-perturbation) is $f_b(x) = y_{true}$.

3.3 Sample-Level Analysis

Post-evasion explanations reveal the direction (*positive*, *negative*, or *neutral*) of each perturbed feature in an adversarial sample x'. Sample-level analysis is performed to empirically assess feature perturbations that positively contribute towards misclassification (*positive perturbations*) against the ones that contribute to maintain the true label as a prediction (*negative perturbations*). Next, we introduce two sample-level metrics which will later serve as foundations to conduct overall correlation analysis over the evasion dataset.

Definition 1: Per-Sample Perturbation Precision (PSPP). Out of all performed feature perturbations ($P(x')$) to produce an adversarial sample x', *PSPP* enables us to compute the rate of perturbations that contribute to change the original prediction y_{true} to another label $y_i \in Y - \{y_{true}\}$. In other words, it measures the rate of perturbed features that are *"negative"* to the original prediction (y_{true}) and *"positive"* to other predictions $y_i \neq y_{true}$. We call such perturbations *positive perturbations* because they positively advance the evasion goal. More formally, the Per-Sample Perturbation Precision for an adversarial sample x' is computed as follows:

$$PSPP(x') = \frac{1}{2}\left(\frac{1}{k-1}\left(\sum_{y_i \in Y\, y_i \neq y_{true}} \frac{pos(x', y_i)}{P(x')}\right) + \frac{neg(x', y_{true})}{P(x')}\right) \quad (1)$$

Equation 1 is the average of two ratios:

- $\left(\frac{1}{k-1}\left(\sum_{y_i \in Y\, y_i \neq y_{true}} \frac{pos(x', y_i)}{P(x')}\right)\right)$: The average rate of perturbed features that are directed to a class $y_i \neq y_{true}$, over all $k-1$ possible false classes $y_i \in Y - \{y_{true}\}$.
- $\left(\frac{neg(x', y_{true})}{P(x')}\right)$: The rate of perturbed features that are not directed to the original label y_{true} and not neutral.

Both ratios that are considered in Eq. 1 measure *Positive Perturbations* that contribute to a misclasssification. We note that $PSPP(x')$ falls in the range $[0, 1]$. The closer $PSPP(x')$ is to 1, the more the overall perturbations performed on the features of x' are precise (effective at feature level). More importantly, when x' evades the model, i.e., $f_b(x') \neq f_b(x)$, then the closer $PSPP(x')$ is to 1 the stronger the correlation between the evasion success and each performed feature perturbation that produced adversarial sample x'.

Definition 2: Per-Sample Perturbation Error (PSPE). Another per-sample measurement for our correlation analysis is the Per-Sample Perturbation Error, $PSPE(x')$, that computes the rate of perturbed features that are directed to the original class y_{true} (*positive* to the original prediction y_{true}). These features stand against the adversary's goal of misclassifying x'. Such features are considered *negative perturbations* with respect to the original class. More formally, $PSPE(x')$ is defined as follows:

$$PSPE(x') = \frac{pos(x', y_{true})}{P(x')} \quad (2)$$

Given an adversarial sample x', $PSPE(x')$ returns the rate of perturbation errors over all perturbed features. We note that a perturbed feature that is *neutral* ($w_j = 0$) to the original prediction ($f_b(x') = y_{true}$) is considered neither as perturbation error nor an effective manipulation to advance the evasion goal. Thus, $PSPE(x')$ may not be directly computed from $PSPP(x')$ and vice versa. Moreover, in the case of a slightly different threat model in-which the evasion is *targeted* to change the original prediction y_{true} to a new target label $y_{target} \in Y - \{y_{true}\}$, then only the term $\frac{pos(x', y_{target})}{P(x')}$ would be considered to compute the perturbation precision $PSPP(x')$, and only the term $\frac{neg(x', y_{target})}{P(x')}$ suffices to compute the rate of committed perturbation errors, $PSPE(x')$.

3.4 Evasion Dataset-Level Analysis

Using $PSPP(x')$ and $PSPE(x')$ defined in Eqs. 1 and 2 as foundations, we now introduce novel correlation analysis metrics that operate at the level of the evasion dataset X'_e to empirically analyze correlation between perturbations and post-evasion explanations.

Definition 3: High-Correlation Rate (HCR). As explained in Sect. 3.3, $PSPP(x')$ quantifies the correlation of each single feature perturbation with the evasion $f_b(x') \neq y_{true}$. The closer $PSPP(x')$ is to 1, the higher is the correlation and vice-versa. We consider a threshold τ that indicates the "strength" of the correlation between positive perturbations on x that resulted in x' and the important features that "explain" $f_b(x') \neq y_{true}$. Based on an empirically estimated τ, we call an adversarial sample x' a *High-Correlated Sample* if $PSPP(x')$ falls in $[\tau, 1]$. In our evaluation, based on empirical observations, we use $\tau = 0.5$.

Based on the above definition, we compute *High-Correlation Rate (HCR)* as the percentage of *High-Correlated Samples* in the evasion set X'_e as follows:

$$HCR = \frac{|X'_e(PSPP > \tau)|}{|X'_e|} \tag{3}$$

where $X'_e(PSPP > \tau) = \{x' \in X'_e : PSPP(x') > \tau\} \cap \{f_b(x') \neq y_{true}\}$. We note that *HCR* quantifies the degree to which adversarial samples are both *evasive* and *correlated* to most feature perturbations performed on original samples.

Definition 4: Average Perturbation Error (APE). As shown in Eq. 2, $PSPE(x')$ computes the number of errors committed during the perturbation of each feature in x to produce the manipulated sample x' (which is the same as computing the number of *negative perturbations*). We leverage $PSPE(x')$ to compute the average of negative perturbations (APE) over all samples in X'_e. Formally, APE is given as follows:

$$APE = \sum_{x' \in X'_e} \frac{PSPE(x')}{|X'_e|} \tag{4}$$

As opposed to aggregate evasion rate that computes the percentage of *evasive samples* versus *non-evasive samples* without deeper insights about the effectiveness of each single feature perturbation, *APE* computes the rate of *"evasive features"* versus *"non-evasive features"* of each evasive sample, over all perturbed samples. Such in-depth investigations into evasion attacks provide a fine-grained assessment of any evasion strategy on ML models.

4 Evaluation

We now evaluate the utility of our suite of metrics for high-fidelity correlation analysis of ML evasion attacks. We describe our datasets and experimental setup in Sect. 4.1 and 4.2, respectively. We then validate our methodology in Sect. 4.3. Finally, we extend our evaluation with a case study in Sect. 4.4.

4.1 Datasets

We use three datasets from two domains. From the malware classification domain, we use two complementary datasets, one based on static analysis, and the other on dynamic (execution behavior) analysis. From image classification, we use a benchmark handwritten digits recognition dataset. We selected these two as representative domains because (*a*) malware detection is a naturally adversarial domain where adversarial robustness to evasion attacks is expected and (*b*) image recognition has been heavily explored for evasion attacks in recent adversarial ML literature. We describe these datasets next.

CuckooTrace (PE Malware). We collected 40K Windows PEs with 50% malware (from VirusShare [4]) and the rest 50% benign PEs (from a goodware site [1]). We use 60% of the dataset as a training set, 25% as a training for explanation substitute model, and the remaining 15% as evasion test. Each sample is represented as a binary feature vector. Each feature indicates the presence/absence of behavioral features captured up on execution of each PE in the Cuckoo Sandbox [2]. Behavioral analysis of 40K PEs resulted in 1549 features, of which 80 are API calls, 559 are I/O system files, and 910 are loaded DLLs.

EMBER (PE Malware). To assess our framework on complementary (static analysis-based) malware dataset, we use EMBER [7], a benchmark dataset of malware and benign PEs released with a trained LightGBM with 97.3% test accuracy. EMBER consists of 2351 features extracted from 1M PEs using a static binary analysis tool LIEF [3]. The training set contains 800K samples composed of 600K labeled samples with 50% split between benign and malicious PEs and 200K unlabeled samples, while the test set consists of 200K samples, again with the same ratio of label split. VirusTotal [5] was used to label all the samples. The feature groups include: PE metadata, header information, byte histogram, byte-entropy histogram, string information, section information, and imported/exported functions. We use 100K of the test set for substitute model training, and the remaining 100K as our evasion set against the LightGBM pretrained model and a DNN which we trained. We use version 2 of EMBER.

MNIST (Image). To further evaluate our framework on image classifiers, we use the MNIST [31] dataset, which comprises 60K training and 10K test images of handwritten digits. The classification task is to identify the digit corresponding to each image. Each 28 × 28 gray-scale sample is encoded as a vector of normalized pixel intensities in the interval [0, 1].

4.2 Models and Setup

Studied ML Models. Across CuckooTrace, EMBER and MNIST, we train 8 models: Multi-Layer Perceptron (MLP), Logistic Regression (LR), Random Forest (RF), Extra Trees (ET), Decision Trees (DT), Light Gradient Boosting decision tree Model (LGBM), a Deep Neural Network (DNN), and a 2D Convolutional Neural Network (CNN). As in prior work [35,36], we choose these models because they are representative of applications of ML across domains including image classification, malware/intrusion detection, and they also complement each other in terms of their architecture and susceptibility to evasion.

Employed Evasion Attacks. Using the evasion set of each dataset, we craft adversarial samples. For the evasion attack, we consider a threat model where the adversary has no knowledge about the target model, but knows features used to train the model (e.g., API calls for malware classifiers, pixels for image classifiers). More precisely, for CuckooTrace and EMBER we incrementally perturb features of a Malware sample until the model flips its label to Benign. Following previous adversarial sample crafting methods [26,49], we adopt only additive manipulations. For instance, for binary features of CuckooTrace (where 1 indicates presence and 0 indicates absence of an API call), we flip only a 0 to 1. Like prior work [46], we also respect the allowable range of perturbations for each static feature in EMBER (e.g., file size is always positive). For MNIST, we add a random noise to the background of the image to change the original gray-scale of each pixel without perturbing white pixels that characterize the handwritten digit. The outcome is an adversarial image that is still recognizable by humans but misclassified by the model. Table 2 shows the comparison between pre-evasion accuracy and post-evasion accuracy. All models exhibit significant drop in the test accuracy after the feature perturbations. We recall that the main purpose of our analysis is to explore the correlation between a perturbed feature and the misclassification result, regardless of the complexity of the evasion strategy. Thus, our choice of perturbation methods is governed by convenience (e.g., execution time) and effectiveness (i.e., results in evasion).

Table 2. Pre-evasion accuracy and post-evasion accuracy across studied models.

Dataset	Model	Pre-evasion accuracy	Post-evasion accuracy	Aggregate evasion accuracy
CuckooTrace	MLP	96%	6.05%	89.95%
CuckooTrace	LR	95%	21.75%	73.25%
CuckooTrace	RF	96%	18.61%	77.39%
CuckooTrace	DT	96%	7.8%	88.20%
CuckooTrace	ET	96%	16.27%	79.73%
EMBER	LGBM	97.3%	56.06%	40.94%
EMBER	DNN	93%	12.08%	80.92%
MNIST	CNN	99.4%	33.1%	66.30%

Table 3. High-correlation rate and average perturbation error for all models.

Dataset	Model	High-correlation rate	Average perturbation error
CuckooTrace	MLP	34.56%	32.96%
CuckooTrace	LR	34.96%	38.92%
CuckooTrace	RF	36.09%	60.73%
CuckooTrace	DT	66.85%	37.86%
CuckooTrace	ET	33.41%	54.62%
EMBER	LGBM	95.03%	7.47%
EMBER	DNN	96.72%	4.89%
MNIST	CNN	44.31%	49.40%

Employed ML Explanation Methods. Informed by recent studies [15,50] that compare the utility of ML explanation methods, we use LIME [39] on CuckooTrace and EMBER, and SHAP [33] on MNIST. More specifically, these studies perform comparative evaluations of black-box ML explanation methods (e.g., LIME [39], SHAP [33], and LEMNA [23]) in terms of effectiveness (e.g., accuracy), stability (i.e., similarity among results of different runs), efficiency (e.g., execution time), and robustness against small feature perturbations. On the one hand, these studies show that LIME performs best on security systems (e.g., Drebin+ [21], Mimicus+ [23]). Thus, we employ LIME on the two malware detection systems (i.e., CuckooTrace and EMBER). On the other hand, SHAP authors proposed a ML explainer called "Deep Explainer", designed for deep learning models, specifically for image classification. Thus, we use SHAP to explain predictions of a CNN on MNIST. We note that independent recent studies [15,50] suggested that both LIME and SHAP outperform LEMNA [23].

4.3 Correlation Analysis Results

Results Overview. Across all models and the three datasets, the evasion attack scores an average $HCR = 55\%$ and $APE = 36\%$. Linking back to what these metrics mean, an average $HCR = 55\%$ shows that for each model an average of only 55% of the adversarial samples have strong feature-level correlation with their respective perturbations. That entails an average of 45% adversarial samples per-model are loosely correlated with their perturbations. APE assesses the per-model average number of negative perturbations per sample. Results in Table 3 suggest a significant rate of negative perturbations are produced by the evasion attack. More precisely, on average across all models around 36% of the perturbations are negative (i.e., they lead the evasion strategy in the wrong direction, by increasing the likelihood of predicting the original label). Next, we expand on these highlights of our findings.

Correlation Between Perturbations and Explanations. Although an evasion attack can achieve a seemingly high aggregate evasion rate (e.g., as high as 94% accuracy drop on MLP on CuckooTrace), we notice that the correlation between each single feature perturbation and a misclassification is not guaranteed. In fact, averaged across models, 45% of the crafted adversarial samples have low-correlated perturbations more than high-correlated ones ($PSPP(x') < 0.5$), suggesting that almost 1 in 2 adversarial samples suffers from weak correlation between post-evasion explanations and pre-evasion perturbations. As a result, counting in such samples in the aggregate evasion rate would essentially give false sense of the effectiveness of an attack strategy at the granularity of each feature perturbation. We underscore that such insights would not have been possible to infer without the high-fidelity correlation analysis. In summary, these results confirm that *not all evasive predictions of an adversarial sample are correlated with the performed feature manipulations.*

Visualizing the Per-sample Perturbation Precision. Figure 3 shows the distribution of $PSPP$ values of all crafted malware samples of CuckooTrace across the 5 models. In the figure, we use the shorthand PP instead of $PSPP$ in the y-axis and we refer to the index of each sample in x-axis. The true prediction of each malware sample is $f_b(x') = 1$, while the evasive prediction is $f_b(x') = 0$ (i.e., adversarial malware sample is misclassified as benign). The red line in the middle represents the threshold τ that decides whether the adversarial sample has more positive perturbations or more negative ones. In almost all the plots, we notice the occurrence of a significant number of low-correlation adversarial malware samples ($PP(x') < \tau$) that evaded the classifier (purple circles below the red line). Once again, these findings suggest that the evasion attack results in a high number of negative perturbations. However, despite the low number of positive perturbations for these samples, the evasion is still successful. This result goes along with our previous finding that high evasion aggregate accuracy can have low correlation with performed perturbations. Thus, even for a successful evasion the perturbations at the feature-level can apparently be ineffective.

It is noteworthy that Fig. 3 exhibits an unusual behavior. Some crafted samples with a true prediction $f_b(x') = 1$ (yellow circles) appear to have high Perturbation Precision, $PP(x') > \tau$, despite the failure to flip the model's prediction from 1 to 0. This is especially true for models: LR, DT, ET and RF. On one hand, this observation suggests that even a small number of negative perturbations ($PP(x') < \tau$) may affect the outcome of the evasion. On the other hand, it suggests potential limitations of Black-Box ML explanation methods in terms of accuracy and stability between different runs. More discussion is provided about this in Sect. 5.

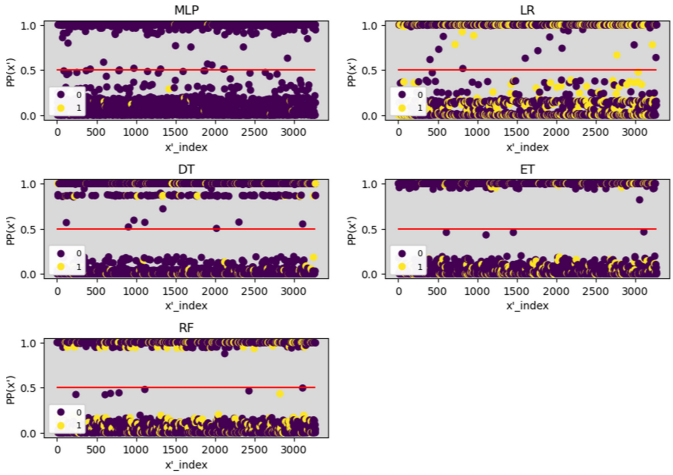

Fig. 3. Distribution of Perturbation Precision (PSPP) values of each adversarial sample across models on CuckooTrace. (Color figure online)

Correlation Analysis Across Domains and Model Architectures. While our results so far strongly suggest the importance of post-evasion correlation analysis for an in-depth assessment of an evasion attack strategy, we also observe that HCR and APE values vary across studied domains (malware, image), model architectures, and feature representations (static, dynamic). This variation speaks to the sensitivity of different domains, models, and feature values to adversarial feature perturbations with implications on robustness and dependability in the face of individual feature perturbation. In fact, ML models trained on EMBER (LGBM and DNN) showed acceptably low rate of negative perturbations (i.e., $APE = 6\%$ on average) and a high rate of samples with highly correlated perturbations (i.e., $HCR = 96\%$ on average). This suggests that static features of Windows PE malware are more sensitive to a feature perturbation considering the higher rate of negative perturbations on dynamic features in CuckooTrace. In terms of comparison between different domains and different ML models, despite the high evasion rate at sample-level, almost all ML models showed some robustness at the level of a single feature perturbation. Most importantly, RF on CuckooTrace showed the highest robustness since more than 60%

of the overall feature perturbations are negative which suggest that they did not contribute to the misclassification decision. ET on CuckooTrace ($APE = 54\%$) and CNN on MNIST ($APE = 49\%$) showed lower robustness than RF, but higher than the other models.

Summary. Our results suggest that aggregate evasion accuracy is inadequate to assess the efficacy of perturbation attack strategy. Our findings also validate that explanation-guided correlation analysis plays a crucial role in diagnosing aggregate evasion rates to winnow high-correlation adversarial samples from low-correlation ones for precise feature-level assessment of evasion accuracy.

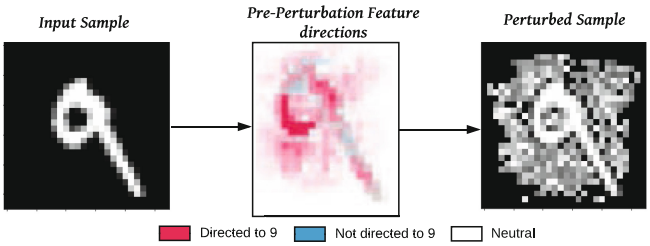

Fig. 4. Example for explanation-guided feature perturbations on MNIST sample. (Color figure online)

4.4 Case Study

The correlation analysis results showed that, while an evasion strategy may result in an evasive adversarial sample, at the granularity of feature perturbations it may produce a considerable number of *negative perturbations*. In other words, the correlation analysis can be leveraged towards more accurate evasion strategy that significantly minimizes negative perturbations. In the following, we explore the potential of explanation methods to guide a more effective evasion strategy.

Explanation-Guided Evasion Strategy. In this case study, we demonstrate how a defender leverages ML explanation methods to examine pre-perturbation predictions before making feature manipulations. In particular, the *pre-perturbation feature directions* reveal *positive features* that significantly contribute to the true prediction (pink pixels in Fig. 4). Intuitively, positive features are strong candidates for perturbations, while *negative features* (blue pixels in Fig. 4) need not be perturbed since they are already directed away from the true label. *Neutral features* (white pixels in Fig. 4) are also not candidates for perturbations since they have no effect on the original label decision. We note that in this case study we consider all positive features (i.e., pink pixels) as candidates for perturbation regardless of the color intensity that represents its explanation weight. In Fig. 4, some positive pixels ($w_i > 0$) with a low explanation weight ($w_i \sim 0$) are almost neutral (i.e., closer to the white color) but still perturbed since they are directed to the true label. Using the same experimental setup, we enhance the evasion strategies used on the three datasets with *explanation-guided pre-perturbation feature selection*. Then, we measure changes to *post-evasion accuracy*, *HCR*, and *APE*.

Impact of Explanation-Guided Evasion. Table 4 suggests an overall improvement not only in aggregate evasion accuracy, but also in the correlation strength between evasion explanations and feature perturbations. Comparing the "Post-Evasion Accuracy" columns of Tables 2 and 4, using explanation-guided evasion strategy post-evasion accuracy drops for all studied models, with an average per-model drop of 13.4% (which translates to the same percentage of improvement in aggregate evasion accuracy). Interestingly, in 4 out of the 5 models in CuckooTrace, post-evasion accuracy drops to zero, with up to 21% drop in post-evasion accuracy for models such as LR. We note that the eventual complete evasion in almost all models in CuckooTrace is most likely attributed to the binary nature of the features, where the explanation-guided feature selection filters out negative features and leaves only positive features that are flipped with just one perturbation. Comparing the HCR columns of Tables 4 and 3, we notice an increase in HCR for all studied models. On average, HCR increased by 27% per-model, which shows the positive utility of the pre-perturbation explanations that guided the evasion strategy to perturb positive features instead of negative ones. Again, comparing the APE columns of Tables 4 and 3, we notice a significant drop in APE, with an average per-model decrease of 20%, which indicates a decrease in the number of *negative perturbations*. Better performance in terms of post-evasion accuracy is also observed for all studied target models.

Table 4. Post-evasion accuracy, HCR, and APE using explanation-guided evasion.

Dataset	Model	Post-evasion accuracy	HCR	APE
CuckooTrace	MLP	0%	92.03%	14.22%
CuckooTrace	LR	0%	96.25%	6%
CuckooTrace	RF	0%	48.53%	51.31%
CuckooTrace	DT	1.41%	98.84%	3.41%
CuckooTrace	ET	0%	48.11%	1.21%
EMBER	LGBM	27.16%	99.58%	2.69%
EMBER	DNN	11.7%	97.8%	2.71%
MNIST	CNN	24.67%	64.5%	43.4%

We note that although we perturb only *positive features*, in the APE column of Table 4 all values are still non-zero. Ideally, the explanation method would guide the perturbation strategy to perform only positive perturbations and make no mistaken perturbations. Nevertheless, we still observe a minimal percentage of *negative perturbations* due to the inherent limitations of the accuracy and stability of explanations by LIME and SHAP, which is also substantiated by recent studies [15,50] that evaluated LIME and SHAP among other ML explanation methods. We will expand on limitations of ML explainers in Sect. 5.

Summary. An explanation-guided feature selection strategy leads to more effective evasion results both in terms of aggregate evasion accuracy and effectiveness at the level of each feature manipulation.

5 Discussion and Limitations

Recent studies [15,50] have systematically compared the performance of ML explanation methods especially on security systems. In addition to general evaluation criteria (e.g., explanation accuracy and sparsity), Warnecke et al. [50] focused on other security-relevant evaluation metrics (e.g., stability, efficiency, and robustness). Fan et al. [15] also proposed a similar framework that led to the same evaluation results. Next, we highlight limitations of LIME [39] and SHAP [33] based on *accuracy* (degree to which relevant features are captured in an explanation), *stability* (how much explanations vary between runs), and *robustness* (the extent to which explanations and prediction are coupled).

Limitations of Explanation Methods. While LIME and SHAP produce more accurate results compared with other black-box explanation methods (e.g., DeepLIFT [44], LEMNA [23]), the *accuracy* of the explanation may vary across different ML model architectures (e.g., MLP, RF, DT, etc.), and across different ML tasks/datasets (e.g., CuckooTrace, EMBER, and MNIST). For instance, the inherent linearity of LIME's approximator could negatively influence its accuracy and stability in explaining predictions of complex models such as RF and ET. More importantly, like all learning-based methods, LIME and SHAP are sensitive to non-determinism (e.g., random initialization, stochastic optimization) which affect their *stability* between different runs. In other words, it is likely to observe a slight variation in the output of multiple runs performed by the same explanation method using the same input data. In fact, we observed that the average difference between Shapley values (i.e., feature importance weights) returned by SHAP is around 1% over 100 runs on the same MNIST sample. Such variation in ML explanation outputs might partly explain some of the unexpected results of our explanation-guided analysis that we noted in Sect. 4.3.

Vulnerability of Explanation Methods. Another issue worth considering is *robustness* of ML explanation methods against adversarial attacks. Studies [19,25,53] have demonstrated that the explanation results are sensitive to small systematic feature perturbations that preserve the predicted label. Such attacks can potentially alter the explanation results, which might in effect influence our explanation-guided analysis. Consequently, our analysis may produce less precise results for correlation metrics such as HCR and APE. Considering the utility of ML explanations we demonstrated in this work, we hope that our framework can be instantiated for adversarial perturbations performed in the problem-space. We note, however, that there needs to be careful consideration in mapping the units of adversarial perturbations in problem space manipulations (e.g., the organ transplant notions proposed in [37]) to the metrics we proposed in Sect. 3. Finally, vulnerability to adversarial attacks is a broader problem in ML, and progress in defense strategies will inspire defense for ML explanation methods.

6 Related Work

Comparing how evasion is assessed, our approach is complementary to prior work which rely on aggregate evasion accuracy. Next, we shed light on prior work focusing on evasion of image classification and malware/intrusion detection.

Image Classification. Several evasion methods have been proposed for image classification tasks. Some of the most notable ones are: Fast Gradient Sign Method (FGSM) [20], Basic Iterative Method (BIM) [30], Projected Gradient Descent (PGD) method [34], and Carlini & Wagner (CW) method [11].

Windows Malware. Kolosnjaji et al. [27] proposed a gradient-based attack against MalConv [38] by appending bytes (in the range 2KB-10KB) to the overlay of a PE. As a follow-up to [27], Demetrio et al. [13] extended the adversarial sample crafting method by demonstrating the feasibility of evasion by manipulating 58 bytes in the DOS header of PE. Suciu et al. [49] explored FGSM [20] to craft adversarial samples against MalConv [27] by padding adversarial payloads between sections in a PE if there is space to perform padding.

Hu and Tan [26] train a substitute model using a GAN to fit a black-box malware detector trained on API call traces. Other works utilize reinforcement learning (RL) to evade malware classifiers. For instance, Anderson et al. [8] aim to limit perturbations to a select set of transformations that are guaranteed to preserve semantic integrity of a sample using RL. Additionally, Apruzzese et al. [9] generate realistic attack samples that evade botnet detectors through deep RL. The generated samples can be used for adversarial learning. Rosenberg et al. [41] adopt the Jacobian-based feature augmentation method introduced in [36] to synthesize training examples to inject fake API calls to PEs at runt-time. Using the augmented dataset, they locally train a substitute model to evade a target RNN black-box malware detector based on API call features.

Android Malware. Like [26], Grosse et al. [21] demonstrate evasion by adding API calls to malicious Android APKs. Yang et al. [52] explore semantic analysis of malicious APKs with the goal of increasing resilience of Android malware detectors against evasion attacks. Recently, Pierazzi et al. [37] take a promising step towards formalization of the mapping between feature space and problem space on Android malware detector.

PDF Malware. Srndic et al. [48] demonstrate vulnerabilities of deployed PDF malware detectors using constrained manipulation with semantic preservation. Xu et al. [51] use genetic algorithms to manipulate ASTs of malicious PDFs to generate adversarial variants while preserving document structure (syntax).

Explanation Methods. In a black-box setting, LIME [39], Anchors [40], SHAP [33], and [22] are among black-box explanation methods that use local approximation. Other methods (e.g., [17,32]) use input perturbation by monitoring prediction deltas. DeepLIFT [44] explains feature importance with respect to a reference output, while white-box explanation techniques (e.g., [45–47]) use gradient-based feature importance estimation. Recent studies [15,19,25,50] explore the

utility of explanation methods across criteria such as accuracy, stability, and robustness of explanations. A more recent interesting application is the use of SHAP signatures to detect adversarial examples in DNNs [16].

In summary, prior work typically rely on comparing pre-perturbation accuracy and post-perturbation accuracy to evaluate ML evasion attacks, which lacks deeper diagnosis of the attack's success. We propose complementary suite of metrics to map a single feature perturbation to its contribution to evasion.

7 Conclusion

We introduced the first explanation-guided methodology for the diagnosis of ML evasion attacks. To do so, we use feature importance-based ML explanation methods to enable high-fidelity correlation analysis between pre-evasion perturbations and post-evasion prediction explanations. To systematize the analysis, we proposed and evaluated a novel suite of metrics. Using image classification and malware detection as representative ML tasks, we demonstrated the utility of the methodology across diverse ML model architectures and feature representations. Through a case study we additionally confirm that our methodology enables evasion attack improvement via pre-evasion feature direction analysis.

Acknowledgements. We thank our shepherd Giovanni Apruzzese and the anonymous reviewers for their insightful feedback that immensely improved this paper.

References

1. CNET freeware site (2020). https://download.cnet.com/s/software/windows/?licenseType=Free
2. Cuckoo sandbox (2020). https://cuckoosandbox.org
3. LIEF project (2020). https://github.com/lief-project/LIEF
4. Virus share (2020). https://virusshare.com
5. Virus total (2020). https://www.virustotal.com/gui/home/upload
6. Ali, A., Eshete, B.: Best-effort adversarial approximation of black-box malware classifiers. In: Park, N., Sun, K., Foresti, S., Butler, K., Saxena, N. (eds.) SecureComm 2020. LNICST, vol. 335, pp. 318–338. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63086-7_18
7. Anderson, H.S., Roth, P.: EMBER: an open dataset for training static PE malware machine learning models. ArXiv e-prints (2018)
8. Anderson, H.S., Kharkar, A., Filar, B., Evans, D., Roth, P.: Learning to evade static PE machine learning malware models via reinforcement learning. CoRR arXiv:1801.08917 (2018)
9. Apruzzese, G., Andreolini, M., Marchetti, M., Venturi, A., Colajanni, M.: Deep reinforcement adversarial learning against botnet evasion attacks. IEEE Trans. Netw. Serv. Manage. **17**, 1975–1987 (2020)
10. Biggio, B., Roli, F.: Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recogn. **84**, 317–331 (2018)
11. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. In: IEEE SP, pp. 39–57 (2017)

12. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. **20**(1), 30–42 (2012)
13. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Explaining vulnerabilities of deep learning to adversarial malware binaries. In: Proceedings of the Third Italian Conference on Cyber Security (2019)
14. Demontis, A., et al.: Yes, machine learning can be more secure! A case study on android malware detection. IEEE TDSC **16**(4), 711–724 (2019)
15. Fan, M., Wei, W., Xie, X., Liu, Y., Guan, X., Liu, T.: Can we trust your explanations? Sanity checks for interpreters in android malware analysis. IEEE Trans. Inf. Forensics Secur. **16**, 838–853 (2021)
16. Fidel, G., Bitton, R., Shabtai, A.: When explainability meets adversarial learning: Detecting adversarial examples using SHAP signatures. In: IEEE IJCNN, pp. 1–8 (2020)
17. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: IEEE ICCV, pp. 3449–3457 (2017)
18. Gao, F., et al.: DeepCC: a novel deep learning-based framework for cancer molecular subtype classification. Oncogenesis **8**(9), 1–12 (2019)
19. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: AAAI, vol. 33 (2017)
20. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
21. Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P.: Adversarial examples for malware detection. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 62–79. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_4
22. Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., Giannotti, F.: Local rule-based explanations of black box decision systems. CoRR arXiv:1805.10820 (2018)
23. Guo, W., Mu, D., Xu, J., Su, P., Wang, G., Xing, X.: LEMNA: explaining deep learning based security applications. In: ACM SIGSAC CCS, pp. 364–379 (2018)
24. Han, D., et al.: Practical traffic-space adversarial attacks on learning-based nidss. CoRR arXiv:2005.07519 (2020)
25. Heo, J., Joo, S., Moon, T.: Fooling neural network interpretations via adversarial model manipulation (2019)
26. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on GAN. CoRR arXiv:1702.05983 (2017)
27. Kolosnjaji, B., et al.: Adversarial malware binaries: evading deep learning for malware detection in executables. In: EUSIPCO, pp. 533–537 (2018)
28. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., Keshet, J.: Deceiving end-to-end deep learning malware detectors using adversarial examples (2018)
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
30. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. CoRR arXiv:1611.01236 (2016)
31. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST database of handwritten digits (2020). http://yann.lecun.com/exdb/mnist/
32. Li, J., Monroe, W., Jurafsky, D.: Understanding neural networks through representation erasure. CoRR arXiv:1612.08220 (2016)
33. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NeurIPS, pp. 4765–4774 (2017)

34. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. CoRR arXiv:1706.06083 (2017)
35. Papernot, N., McDaniel, P.D., Goodfellow, I.J.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR arXiv:1605.07277 (2016)
36. Papernot, N., McDaniel, P.D., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. CoRR arXiv:1602.02697 (2016)
37. Pierazzi, F., Pendlebury, F., Cortellazzi, J., Cavallaro, L.: Intriguing properties of adversarial ML attacks in the problem space. In: IEEE SP (2020)
38. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole EXE. In: AAAI Workshops, pp. 268–276 (2018)
39. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": explaining the predictions of any classifier. In: ACM SIGKDD, pp. 1135–1144 (2016)
40. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: high-precision model-agnostic explanations. In: AAAI, pp. 1527–1535 (2018)
41. Rosenberg, I., Shabtai, A., Rokach, L., Elovici, Y.: Generic black-box end-to-end attack against state of the art API call based malware classifiers. In: Bailey, M., Holz, T., Stamatogiannakis, M., Ioannidis, S. (eds.) RAID 2018. LNCS, vol. 11050, pp. 490–510. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00470-5_23
42. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.K.: Deep reinforcement learning framework for autonomous driving. CoRR arXiv:1704.02532 (2017)
43. Shapley, L.: A value for n-person games (1953)
44. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: ICML, pp. 3145–3153 (2017)
45. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. In: ICLR Workshop Track Proceedings (2014)
46. Smilkov, D., Thorat, N., Kim, B., Viégas, F.B., Wattenberg, M.: SmoothGrad: removing noise by adding noise. CoRR arXiv:1706.03825 (2017)
47. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: the all convolutional net. In: ICLR Workshop Track Proceedings (2015)
48. Srndic, N., Laskov, P.: Practical evasion of a learning-based classifier: a case study. In: IEEE SP, pp. 197–211 (2014)
49. Suciu, O., Coull, S.E., Johns, J.: Exploring adversarial examples in malware detection. In: IEEE SP Workshops, pp. 8–14 (2019)
50. Warnecke, A., Arp, D., Wressnegger, C., Rieck, K.: Evaluating explanation methods for deep learning in security. In: IEEE EuroSP, pp. 158–174 (2020)
51. Xu, W., Qi, Y., Evans, D.: Automatically evading classifiers: a case study on PDF malware classifiers. In: NDSS (2016)
52. Yang, W., Kong, D., Xie, T., Gunter, C.A.: Malware detection in adversarial settings: exploiting feature evolutions and confusions in android apps. In: ACSAC, pp. 288–302 (2017)
53. Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., Wang, T.: Interpretable deep learning under fire. In: USENIX Security, pp. 1659–1676 (2020)